SMART CONTRACT AUDIT

ZOKYO.

September 6th, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges

TECHNICAL SUMMARY

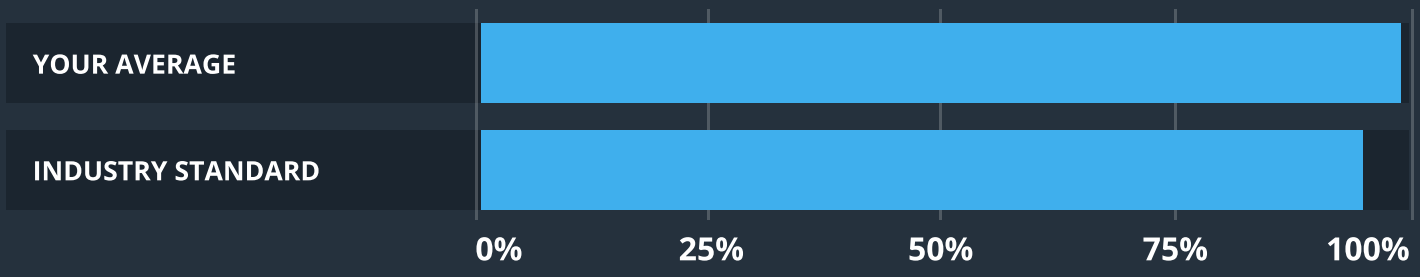
This document outlines the overall security of the Propel smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Propel smart contract codebase for quality, security, and correctness.

Contract Status

There was 1 critical issues found during the audit.

Testable Code



The testable code is 99.13%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Propel team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Propel repository.

Repository:
https://github.com/PropelXYZ/PropelPoolContracts/commit/efdba5fb5c51d61c9de710f2dccfee82050fbeff

Last commit:
30810b3678998ea858461215fbad7e20445ff80c

Contracts:

- PropelFactory.sol
- PropelLpPool.sol
- LpPoolConfig.sol
- CloneOwnable.sol
- SafePropelERC20.sol

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.
2	Cross-comparison with other, similar smart contracts by industry leaders.
3	Testing contract logic against common and uncommon attack vectors.
4	Thorough, manual review of the codebase, line-by-line.

SUMMARY

There was 1 critical issue found during the audit which was successfully resolved by the Propel team. Besides the critical issue, several findings were found as well. They are presented below in the “Complete Analysis” section. Among them are 2 issues with high severity and one issue with medium severity.

All the mentioned findings may have an effect only in case of specific conditions. It is worth mentioning that all of the issues were resolved. All the findings with critical, high, and medium severity were fixed and bear no risk for the user

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

Incorrect behaviour of stake method at PropelLpPool.sol

CRITICAL | RESOLVED

The error appeared in the last commit. In the function delegateStake in contract PropelFactory set parameter of the function Stake msgSender(), but in the contract PropelLpPool is written to the event of function Stake address of contract PropelFactory.

Recommendation:
Use parameter _sender in function Stake.

Re-audit:
Fixed.

Unnecessary _id parameter at functions

HIGH | RESOLVED

Different contracts in different functions have parameter _id. This parameter doesn't influence any contract logic and needed only for FE purpose. But these functions can be called for example from other contracts (from different platforms). In this situation _id parameter is useless.

Recommendation:
There are few possible solutions for this:

- remove this parameter from contract;
- allow to make transactions only by person addresses or whitelisted contracts (in this way you will be restrict potential clients);
- add comment in code to explain why this variable is existing.

Re-audit:
Fixed, added comments.

Staking via factory

HIGH | RESOLVED

Staking to the pool happening via factory but unstake/withdraw via pool. It may lead to confusion in usage.

Recommendation:
There are few possible solutions for this:

- make all three (stake, unstake, and withdraw) function happening in one contract for client;
- leave comment in code to explain why staking should be done through factory.

Misleading comment at createPool in PropelFactory contract

MEDIUM | RESOLVED

There is a comment: "to create a dynamic pool keep _totalRewardAmount as 0", but then in code there is require:

```
require(_totalRewardAmount > 0, "Invalid totalRewardAmount");
```

Recommendation:
Either fix a comment either remove require.

Re-audit:
Fixed, added comments.

Additional check are required for stake in PropelLpPool contract

LOW | RESOLVED

Recommendation:
Add check for _amount to be greater than 0.

Re-audit:
They added require.

Unnecessary require at createPool in PropelFactory contract

LOW | RESOLVED

In case when you want to activate pool immediately _startAfter can be 0.

```
require(_startAfter > 0, "Invalid start");
```

Recommendation:
Remove line 72.

Re-audit:
They will not start pool immediately.

Misleading comment at isStakingStarted at PersonalLpPool contract

LOW | RESOLVED

It have two requires:

```
require(block.number >= startBlock, "Staking has not started yet");
require(block.number <= endBlock, "Staking has already closed");
```

But if the current block.number is sold then the ending block means that we still have time. And same with startBlock.

Recommendation:
Fix comments or expressions.

Re-audit:
Fixed.

No checks in init at PropelLpPool contract

LOW | NOT VALID

Init method can get endblock > startblock values, or rewardVault can be zero address (it can be fine if pool is compounding).

Recommendation:
Add following requires:

- endblock > startblock;
- totalRewardAmount > 0;
- rewardVault != address(0) (can be scipet in case of compound pool).

Re-audit:
Checks already mades in factory.

If can be simplified in closeSubPool at PropelLpPool contract

INFORMATIONAL | RESOLVED

```
if (!subPoolClosed) {
        subPoolClosed = true;
}
```

It can be written as:

```
subPoolClosed = true
```

Recommendation:
Simplify if expression.

Re-audit:
Logic are changed, so this issues are fixed.

Direct msg.sender in CloneOwnable contract

INFORMATIONAL | RESOLVED

CloneOwnable inherits Context contract, so can be used _msgSender function.

Recommendation:
Use _msgSender() instead of msg.sender.

Re-audit:
Fixed.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Propel team

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
common\	88.89	62.50	80.00	81.82	
CloneOwnable.sol	88.89	62.50	80.00	81.82	34, 35
factory\	100.00	83.33	100.00	100.00	
PropelFactory.sol	100.00	83.33	100.00	100.00	
interfaces\	100.00	100.00	100.00	100.00	
IERC2612Permit.sol	100.00	100.00	100.00	100.00	
IFactory.sol	100.00	100.00	100.00	100.00	
ILpPoll.sol	100.00	100.00	100.00	100.00	
IPropelERC20.sol	100.00	100.00	100.00	100.00	
IRewardVault.sol	100.00	100.00	100.00	100.00	
libraries\	55.56	33.33	50.00	50.00	
SafePropelERC20.sol	55.56	33.33	50.00	50.00	51, 55, 63, 64, 72
lpPool\	95.77	93.75	93.75	95.81	
LPPoolConfig.sol	100.00	100.00	100.00	100.00	
PropelLpPool.sol	95.77	93.75	93.75	95.81	... 592, 593, 595
mockTokens\	100.00	100.00	100.00	100.00	
ERC20Mock.sol	100.00	100.00	100.00	100.00	
rewardVault\	100.00	100.00	100.00	100.00	

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
PropelRewardVault.sol	100.00	100.00	100.00	100.00	
token\	0.00	0.00	0.00	0.00	
ERC20Permit.sol	0.00	0.00	0.00	0.00	... 59, 61, 62, 69
PropelToken.sol	0.00	100.00	0.00	0.00	23, 35, 36
All files	**90.11**	**85.94**	**83.58**	**89.3**	

Tests written by Zokyo Security team

As part of our work assisting Propel in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Propel contract requirements for details about issuance amounts and how the system handles these.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
access\	100.00	87.50	100.00	100.00	
CloneOwnable.sol	100.00	87.50	100.00	100.00	
factory\	100.00	100.00	100.00	100.00	
PropelFactory.sol	100.00	100.00	100.00	100.00	
libraries\	100.00	66.67	100.00	100.00	
SafeERC20.sol	100.00	66.67	100.00	100.00	
lpPool\	98.93	98.89	100.00	98.94	
LpPoolConfig.sol	100.00	100.00	100.00	100.00	
PropelLpPool.sol	98.93	98.89	100.00	98.94	
All files	**99.13**	**95.55**	**100.00**	**99.15**	

We are grateful to have been given the opportunity to work with the Propel team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the Propel team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.